NetServ: Deploying Customized Network Services on Demand

Henning Schulzrinne,

Jae Woo Lee & Suman Srinivasan

Columbia University

Joint work with: Bell Labs (Alcatel-Lucent), Deutsche Telekom, DOCOMO Euro-Labs

NetServ overview

Extensible architecture for core network services

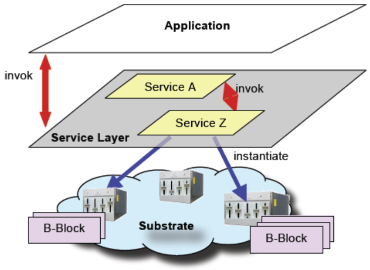


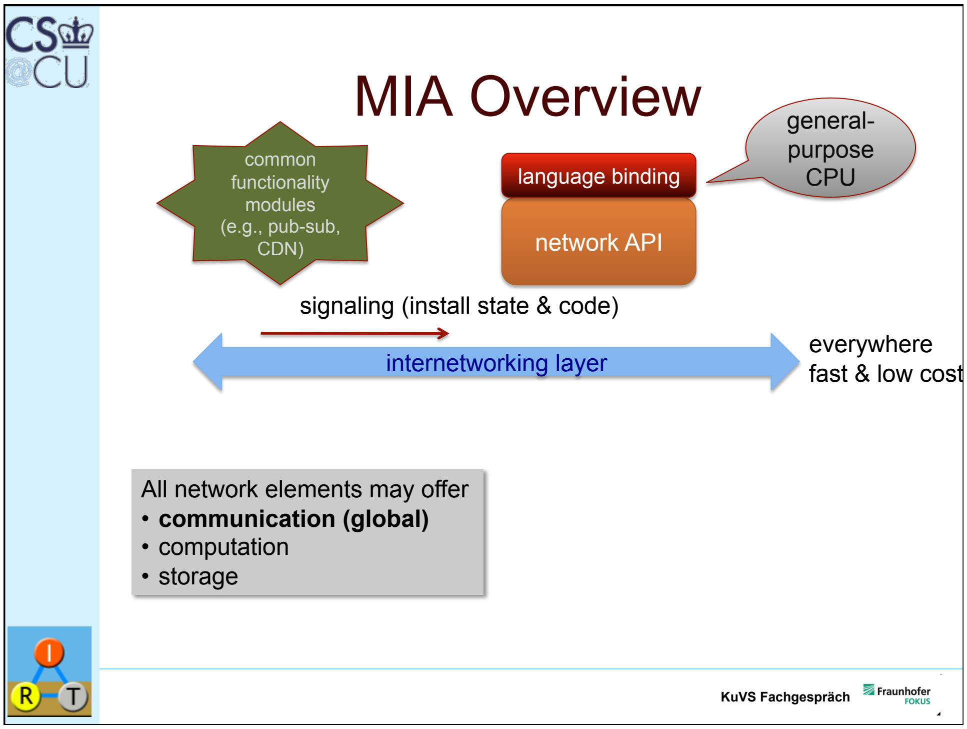
Figure 3: Instantiation of services over tunable building blocks.

Modularization
- Building Blocks
- Service Modules

Virtual services framework
- Security
- Portability

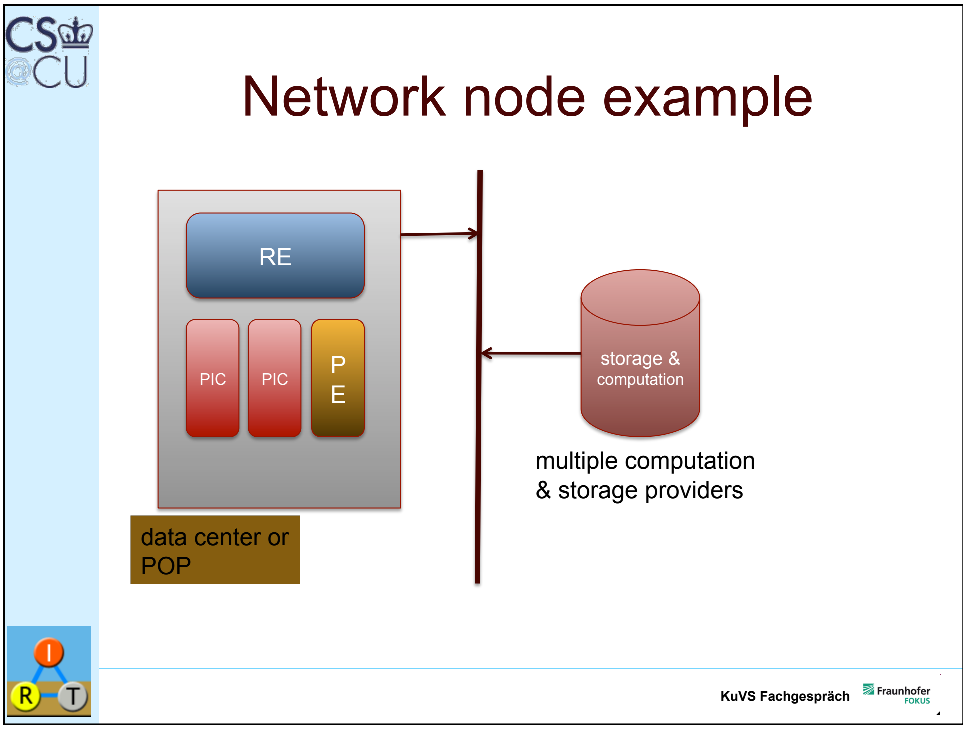
NSF FIND four-year project
- Columbia University
- Bell Labs
- Deutsche Telekom
- DOCOMO Euro-Labs

No more *ossification* in NGI



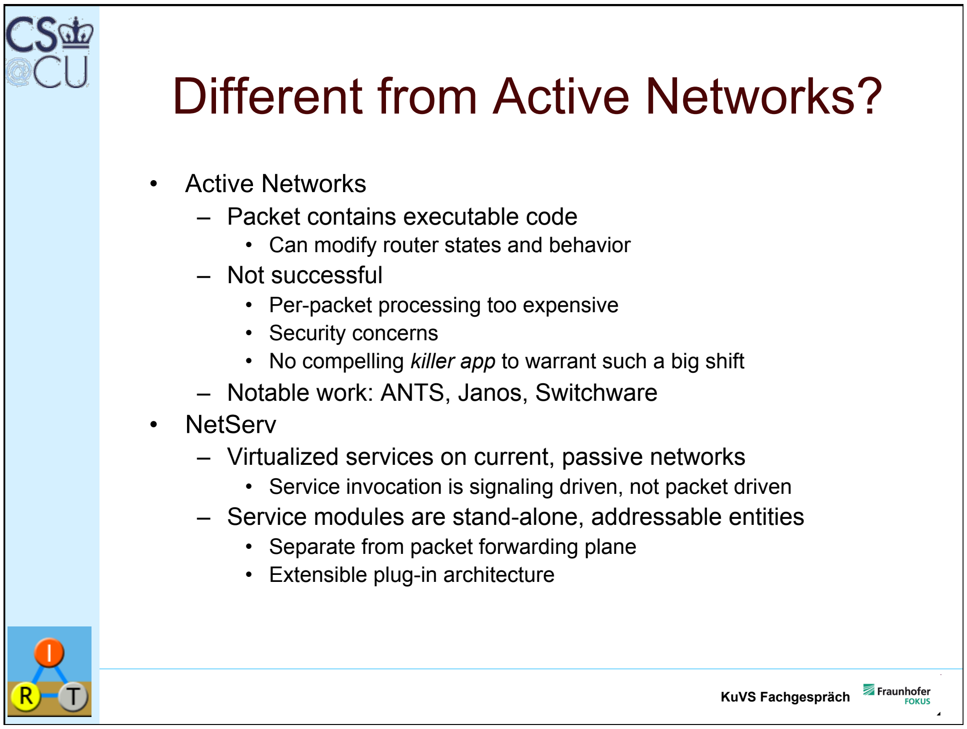
MIA Overview

All network elements may offer
- **communication (global)**
- computation
- storage



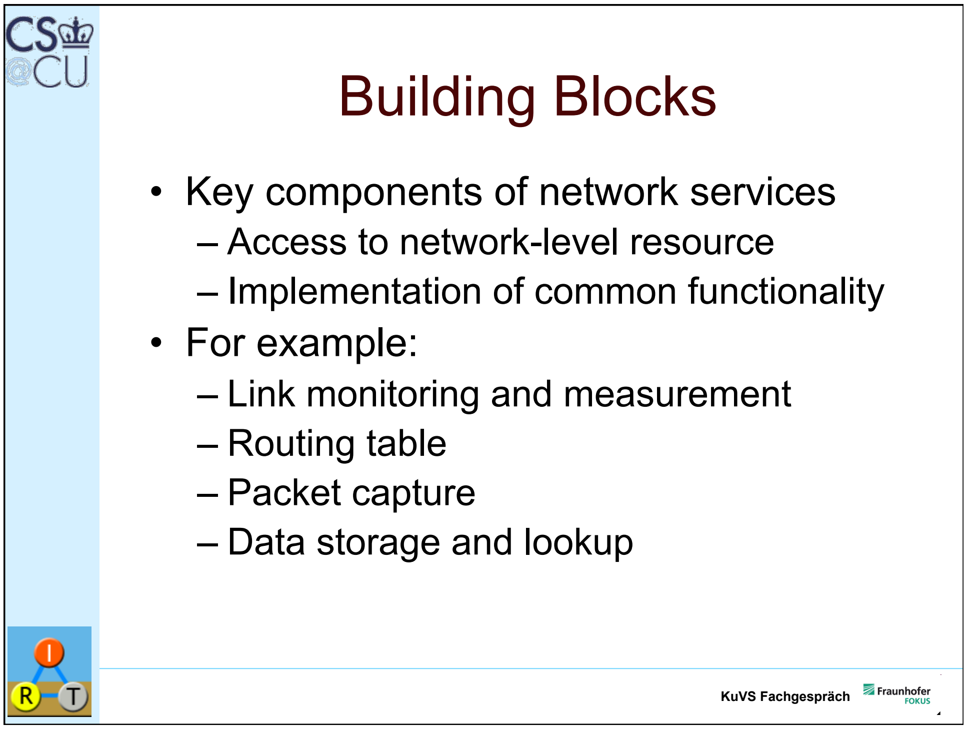
Network node example

multiple computation & storage providers



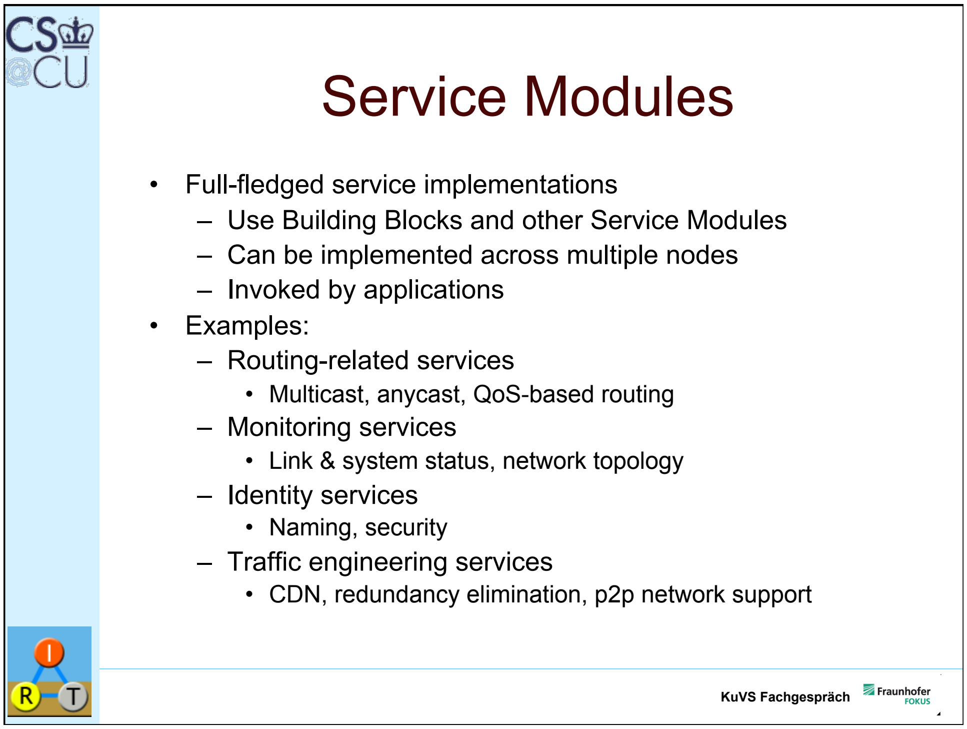
Different from Active Networks?

- Active Networks
  - Packet contains executable code
    - Can modify router states and behavior
  - Not successful
    - Per-packet processing too expensive
    - Security concerns
    - No compelling *killer app* to warrant such a big shift
  - Notable work: ANTS, Janos, Switchware
- NetServ
  - Virtualized services on current, passive networks
    - Service invocation is signaling driven, not packet driven
  - Service modules are stand-alone, addressable entities
    - Separate from packet forwarding plane
    - Extensible plug-in architecture



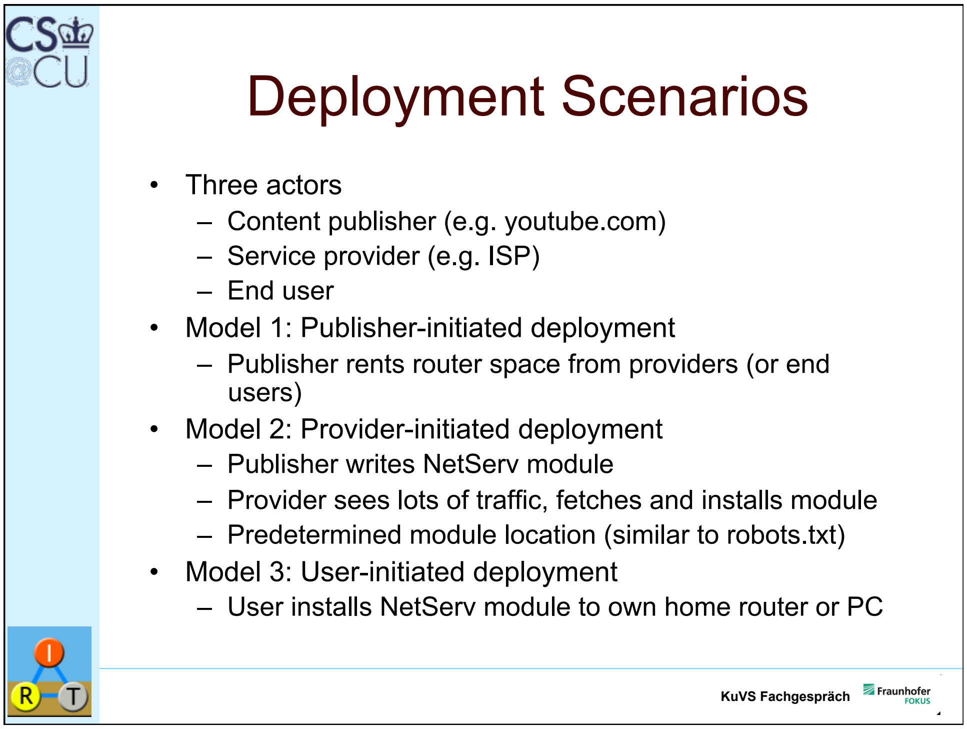
Building Blocks

- Key components of network services
  - Access to network-level resource
  - Implementation of common functionality
- For example:
  - Link monitoring and measurement
  - Routing table
  - Packet capture
  - Data storage and lookup



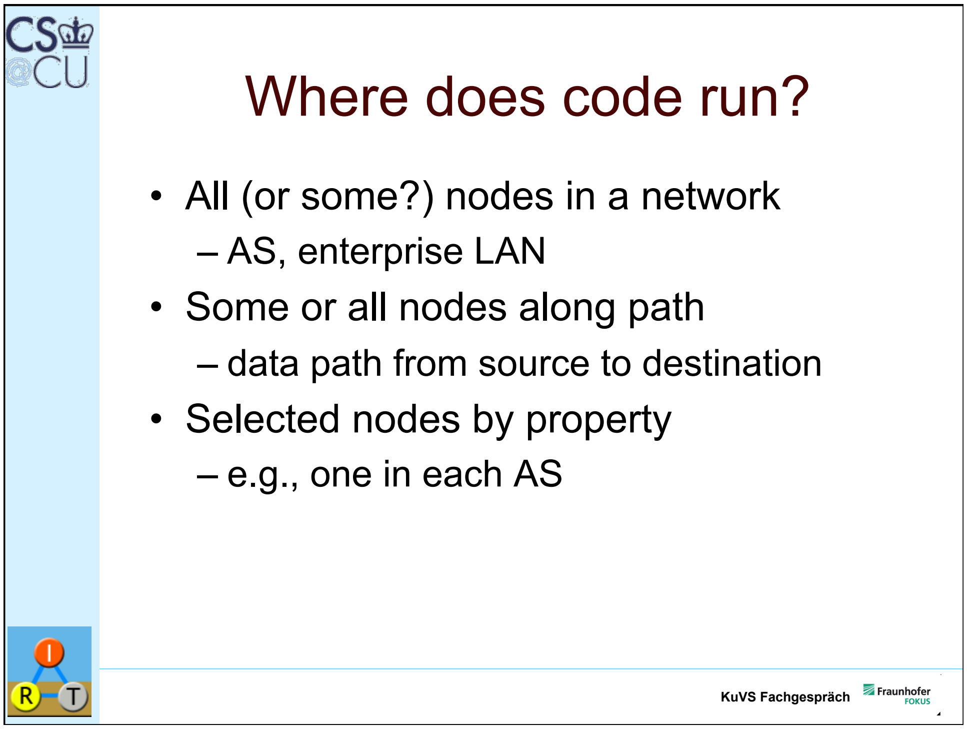
Service Modules

- Full-fledged service implementations
  - Use Building Blocks and other Service Modules
  - Can be implemented across multiple nodes
  - Invoked by applications
- Examples:
  - Routing-related services
    - Multicast, anycast, QoS-based routing
  - Monitoring services
    - Link & system status, network topology
  - Identity services
    - Naming, security
  - Traffic engineering services
    - CDN, redundancy elimination, p2p network support



Deployment Scenarios

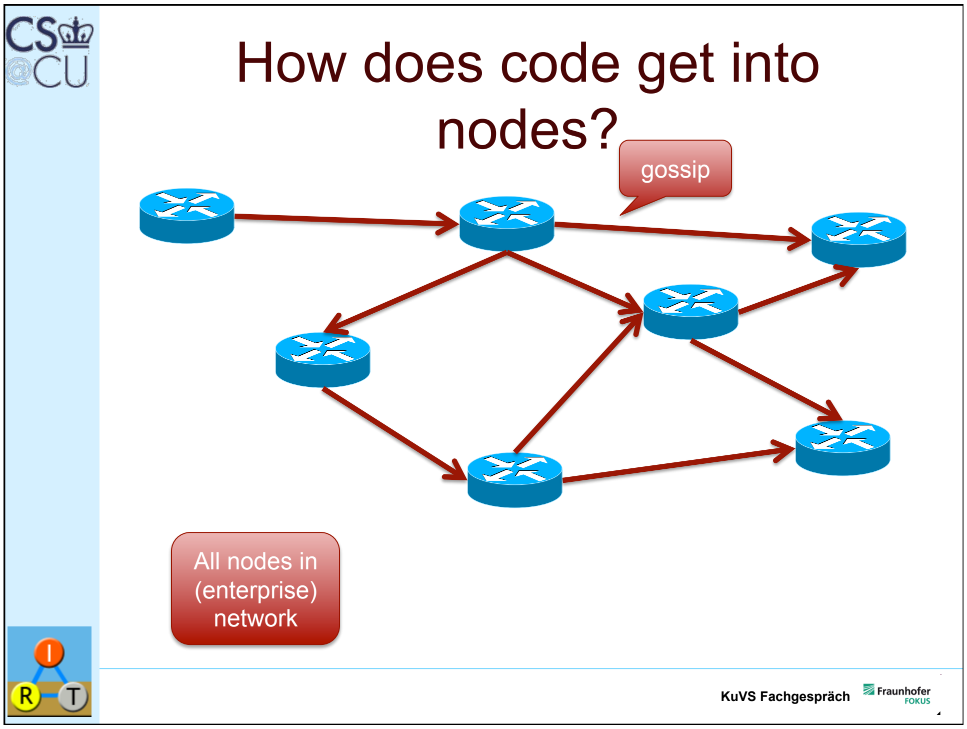
- Three actors
  - Content publisher (e.g. youtube.com)
  - Service provider (e.g. ISP)
  - End user
- Model 1: Publisher-initiated deployment
  - Publisher rents router space from providers (or end users)
- Model 2: Provider-initiated deployment
  - Publisher writes NetServ module
  - Provider sees lots of traffic, fetches and installs module
  - Predetermined module location (similar to robots.txt)
- Model 3: User-initiated deployment
  - User installs NetServ module to own home router or PC



Where does code run?

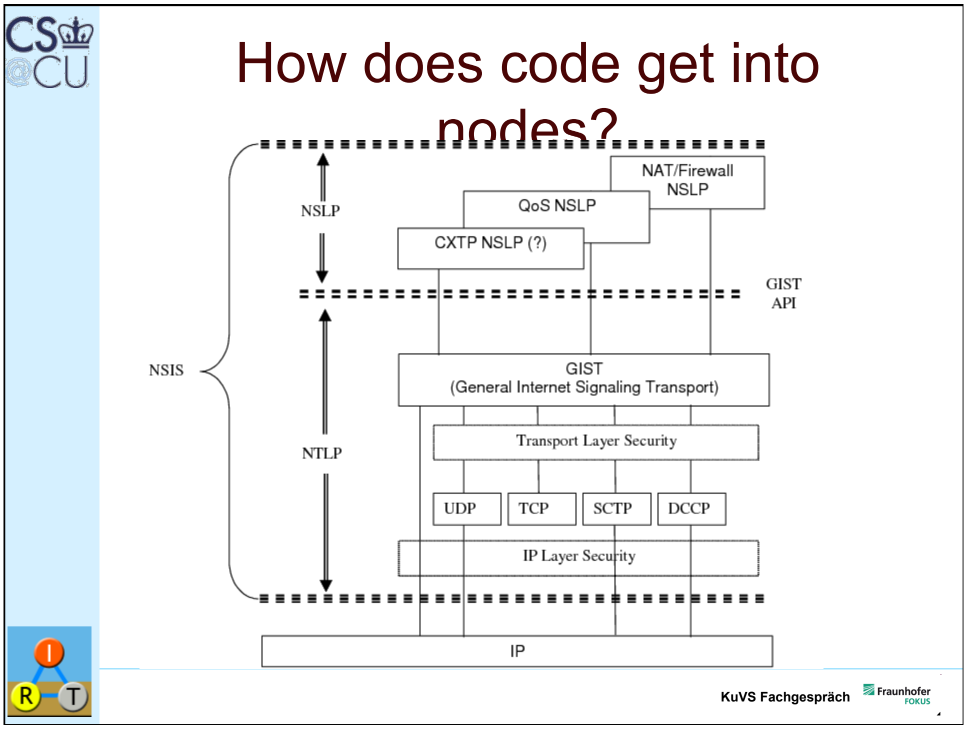
- All (or some?) nodes in a network
  - AS, enterprise LAN
- Some or all nodes along path
  - data path from source to destination
- Selected nodes by property
  - e.g., one in each AS

KuVS Fachgespräch

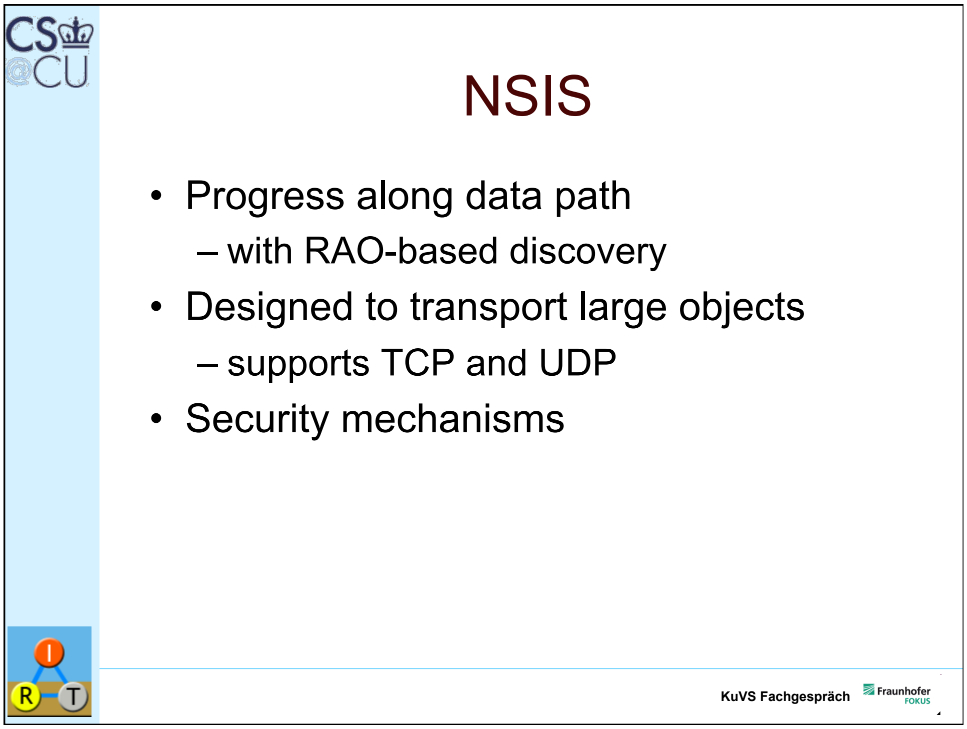


How does code get into nodes?

KuVS Fachgespräch

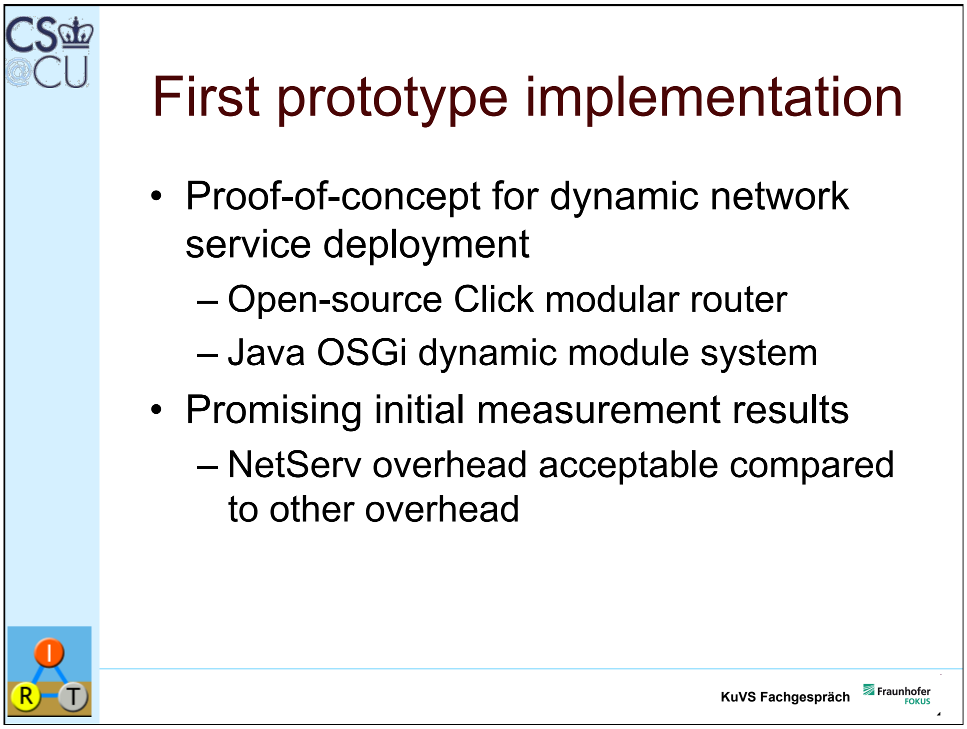


How does code get into nodes?



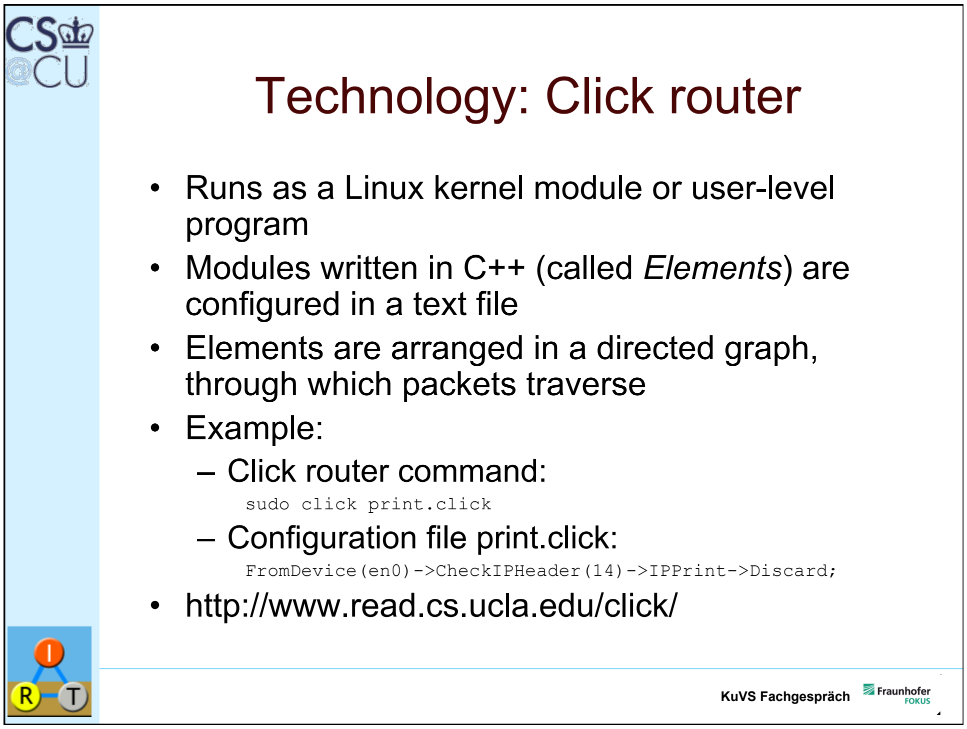
NSIS

- Progress along data path
  - with RAO-based discovery
- Designed to transport large objects
  - supports TCP and UDP
- Security mechanisms



First prototype implementation

- Proof-of-concept for dynamic network service deployment
  - Open-source Click modular router
  - Java OSGi dynamic module system
- Promising initial measurement results
  - NetServ overhead acceptable compared to other overhead



Technology: Click router

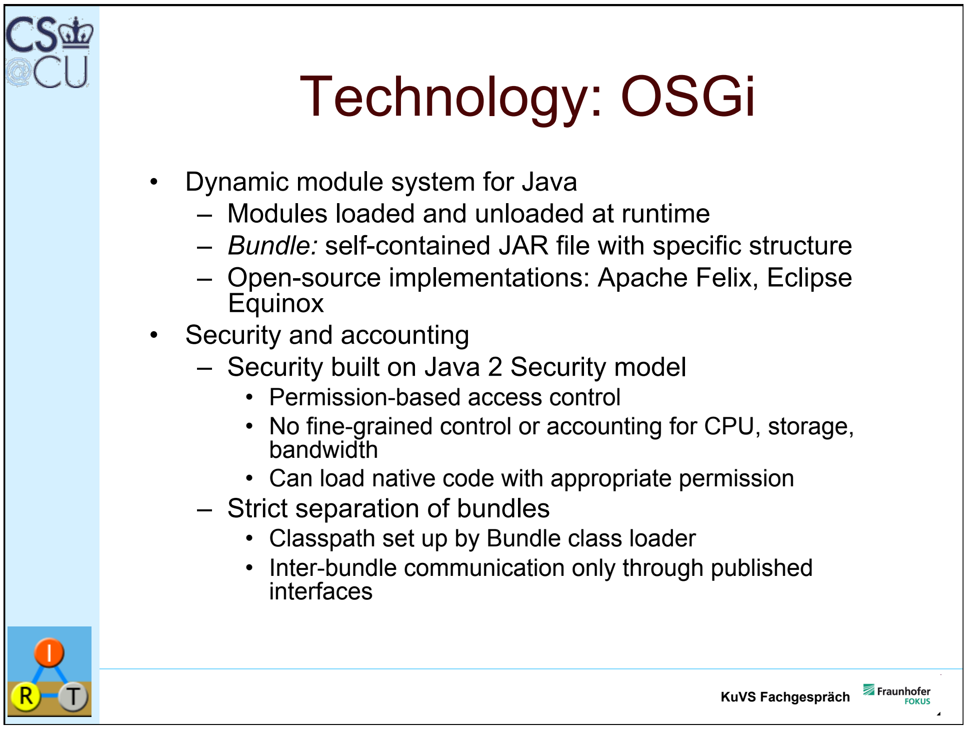
- Runs as a Linux kernel module or user-level program
- Modules written in C++ (called *Elements*) are configured in a text file
- Elements are arranged in a directed graph, through which packets traverse
- Example:
  - Click router command:

```
sudo click print.click
```

  - Configuration file print.click:

```
FromDevice(en0)->CheckIPHeader(14)->IPPrint->Discard;
```

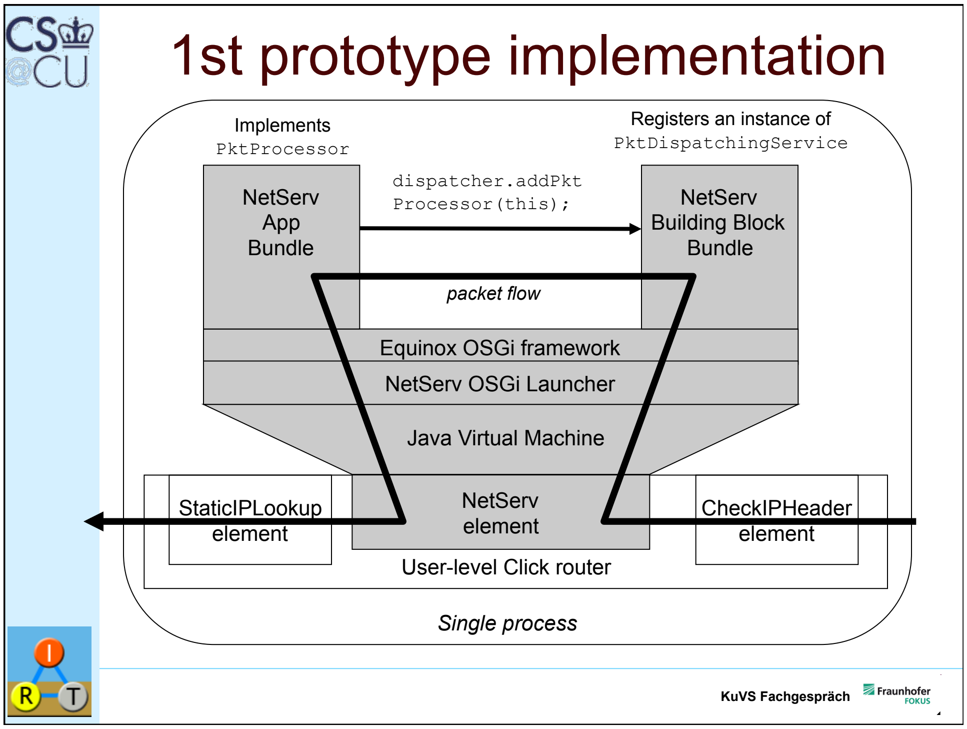
- http://www.read.cs.ucla.edu/click/



Technology: OSGi

- Dynamic module system for Java
  - Modules loaded and unloaded at runtime
  - *Bundle:* self-contained JAR file with specific structure
  - Open-source implementations: Apache Felix, Eclipse Equinox
- Security and accounting
  - Security built on Java 2 Security model
    - Permission-based access control
    - No fine-grained control or accounting for CPU, storage, bandwidth
    - Can load native code with appropriate permission
  - Strict separation of bundles
    - Classpath set up by Bundle class loader
    - Inter-bundle communication only through published interfaces

KuVS Fachgespräch



1st prototype implementation

Implements `PktProcessor`

Registers an instance of `PktDispatchingService`

NetServ App Bundle

`dispatcher.addPkt Processor(this);`

NetServ Building Block Bundle

packet flow

Equinox OSGi framework

NetServ OSGi Launcher

Java Virtual Machine

StaticIPLookup element

NetServ element

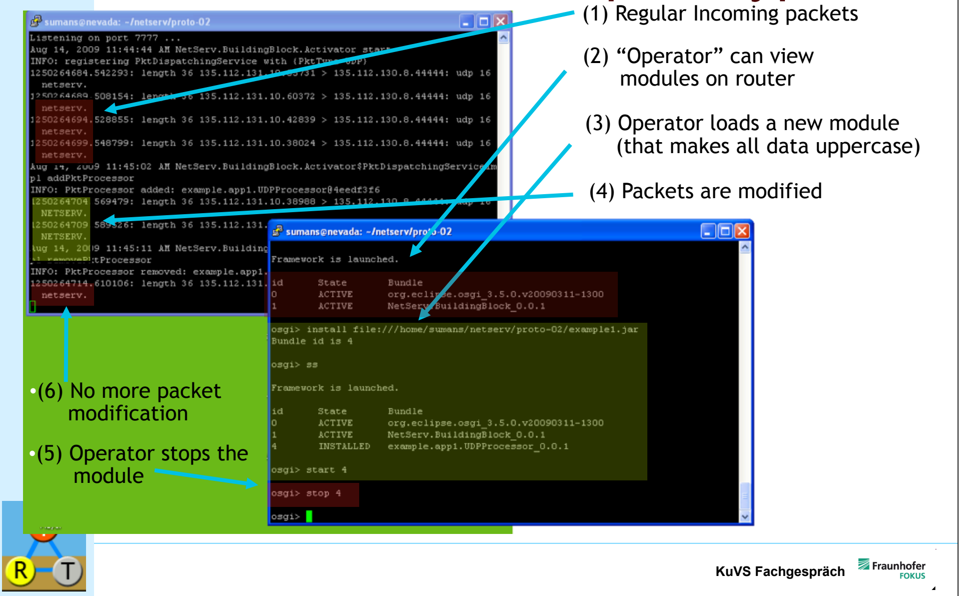
CheckIPHeader element

User-level Click router

Single process

KuVS Fachgespräch

Demo: NetServ prototype



```
sumans@nevada: ~/netserv/proto-02
Listening on port 7777 ...
Aug 14, 2009 11:44:44 AM NetServ.BuildingBlock.Activator start
INFO: registering PktDispatchingService with (PktType=UDP)
1250264684.542293: length 36 135.112.131.10.[illegible]731 > 135.112.130.8.44444: udp 16
  netserv.
1250264689.508154: length 36 135.112.131.10.60372 > 135.112.130.8.44444: udp 16
  netserv.
1250264694.528855: length 36 135.112.131.10.42839 > 135.112.130.8.44444: udp 16
  netserv.
1250264699.548799: length 36 135.112.131.10.38024 > 135.112.130.8.44444: udp 16
  netserv.
Aug 14, 2009 11:45:02 AM NetServ.BuildingBlock.Activator$PktDispatchingServiceIm
pl addPktProcessor
INFO: PktProcessor added: example.app1.UDPProcessor@4eedf3f6
1250264704.569479: length 36 135.112.131.10.38988 > 135.112.130.8.44444: udp 16
  NETSERV.
1250264709.589326: length 36 135.112.131.
  NETSERV.
Aug 14, 2009 11:45:11 AM NetServ.Building
pl removePktProcessor
INFO: PktProcessor removed: example.app1.
1250264714.610106: length 36 135.112.131.
  netserv.
```

```
sumans@nevada: ~/netserv/proto-02

Framework is launched.

id      State       Bundle
0       ACTIVE      org.eclipse.osgi_3.5.0.v20090311-1300
1       ACTIVE      NetServ.BuildingBlock_0.0.1

osgi> install file:///home/sumans/netserv/proto-02/example1.jar
Bundle id is 4

osgi> ss

Framework is launched.

id      State       Bundle
0       ACTIVE      org.eclipse.osgi_3.5.0.v20090311-1300
1       ACTIVE      NetServ.BuildingBlock_0.0.1
4       INSTALLED   example.app1.UDPProcessor_0.0.1

osgi> start 4

osgi> stop 4

osgi>
```

(1) Regular Incoming packets

(2) “Operator” can view modules on router

(3) Operator loads a new module (that makes all data uppercase)

(4) Packets are modified

- (5) Operator stops the module
- (6) No more packet modification

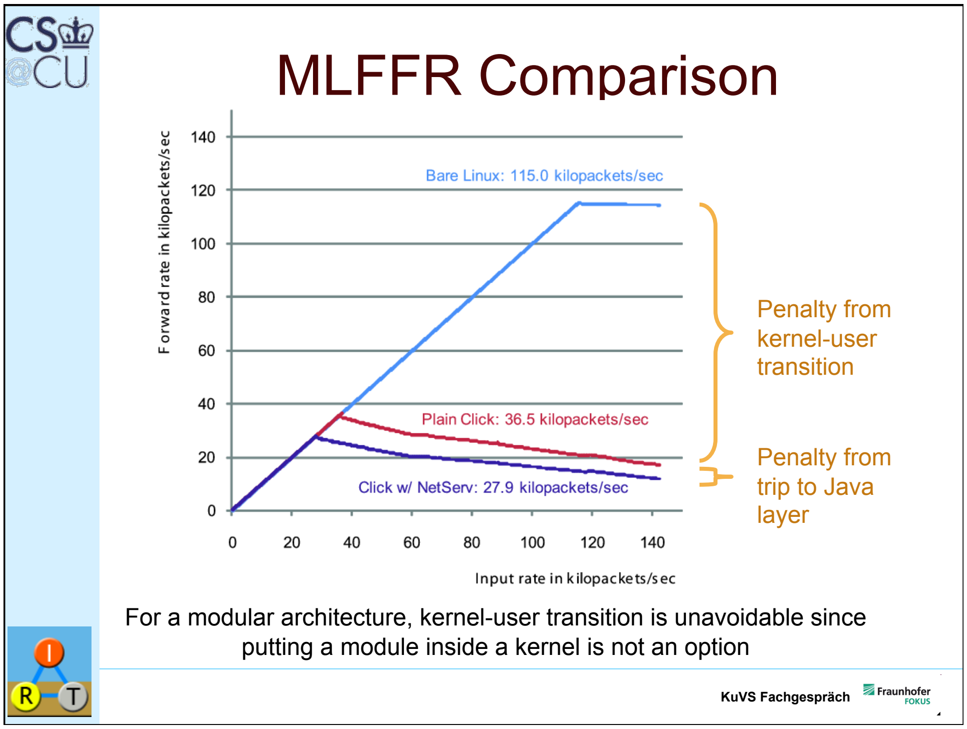
KuVS Fachgespräch

Performance Evaluation

- Initial measurements on the first prototype
  - NetServ on user-level Click router
  - Maximum Loss Free Forward Rate (MLFFR)
- Future work on next-generation prototypes
  - NetServ on JUNOS, kernel-mode Click
  - Ping latency
  - Microbenchmarks
  - Throughput for non-trivial services

18

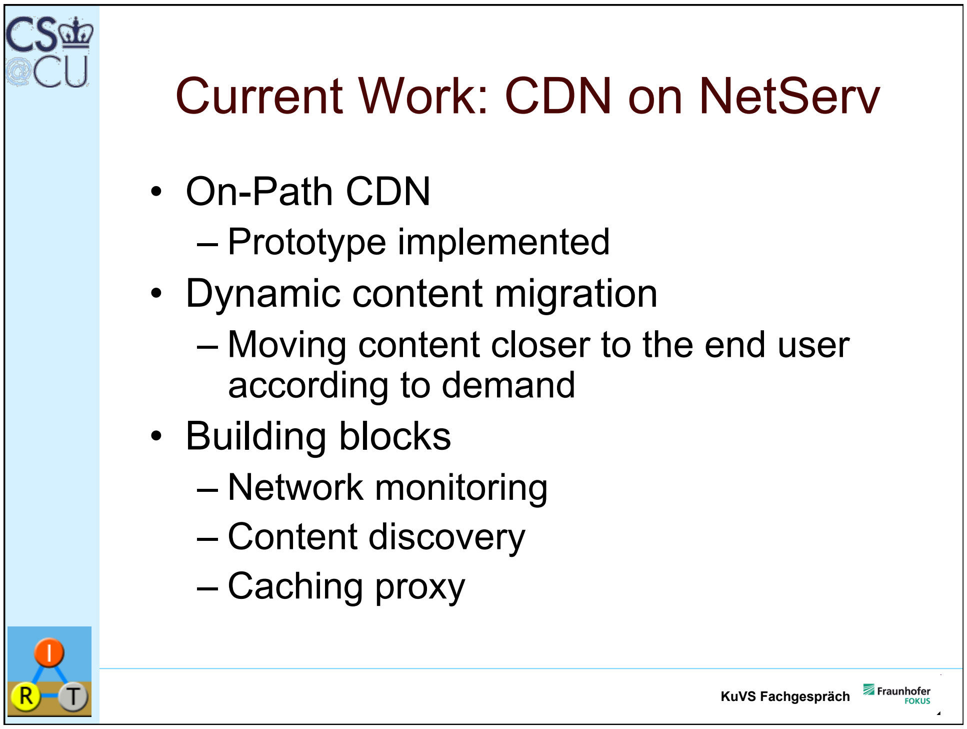
KuVS Fachgespräch



MLFFR Comparison

For a modular architecture, kernel-user transition is unavoidable since putting a module inside a kernel is not an option

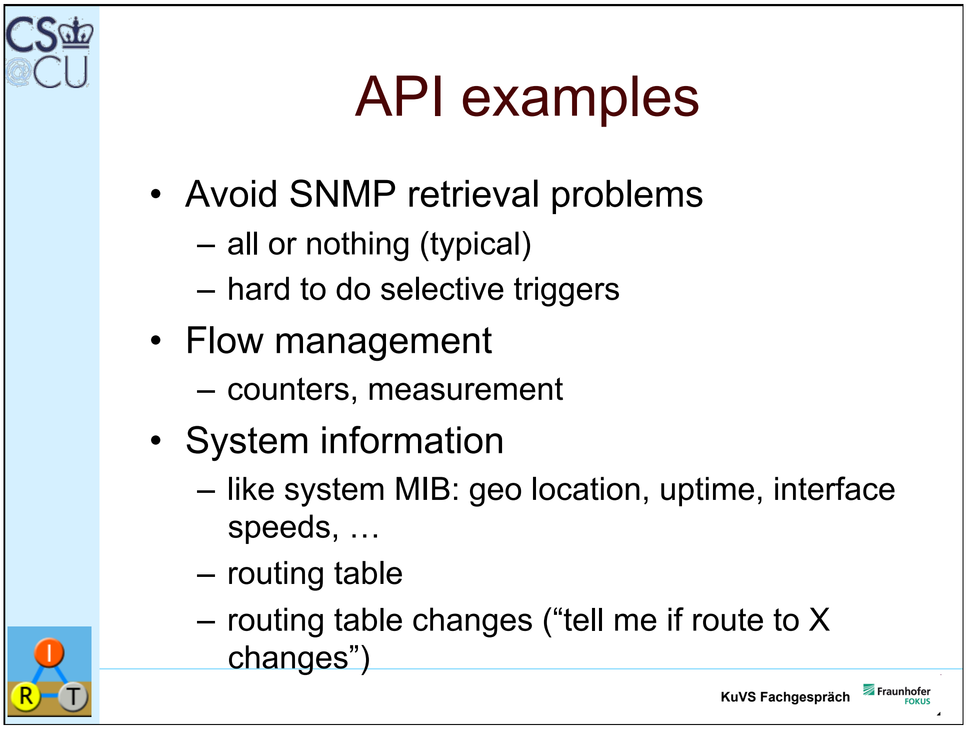
KuVS Fachgespräch



Current Work: CDN on NetServ

- On-Path CDN
  - Prototype implemented
- Dynamic content migration
  - Moving content closer to the end user according to demand
- Building blocks
  - Network monitoring
  - Content discovery
  - Caching proxy

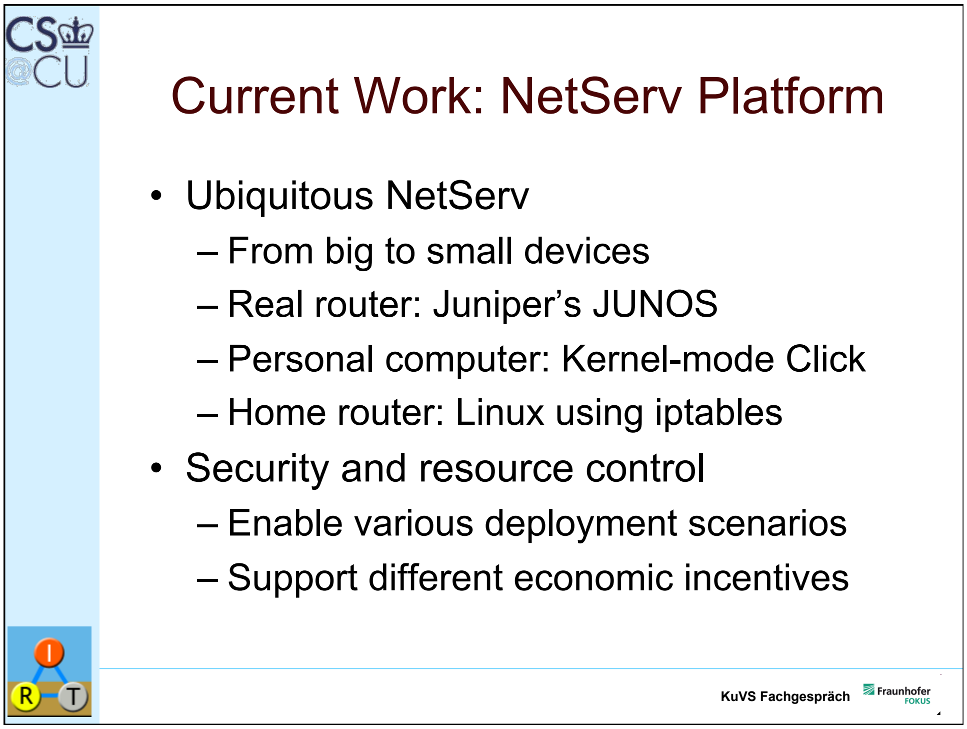
KuVS Fachgespräch



API examples

- Avoid SNMP retrieval problems
  - all or nothing (typical)
  - hard to do selective triggers
- Flow management
  - counters, measurement
- System information
  - like system MIB: geo location, uptime, interface speeds, …
  - routing table
  - routing table changes (“tell me if route to X changes”)

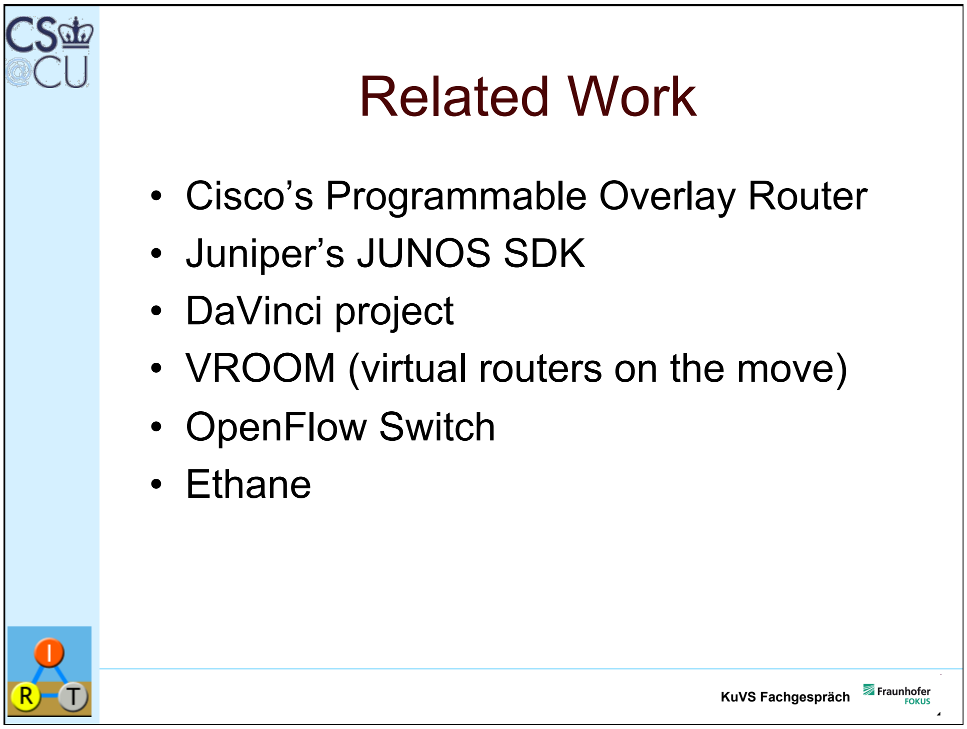
KuVS Fachgespräch Fraunhofer FOKUS



Current Work: NetServ Platform

- Ubiquitous NetServ
  - From big to small devices
  - Real router: Juniper’s JUNOS
  - Personal computer: Kernel-mode Click
  - Home router: Linux using iptables
- Security and resource control
  - Enable various deployment scenarios
  - Support different economic incentives

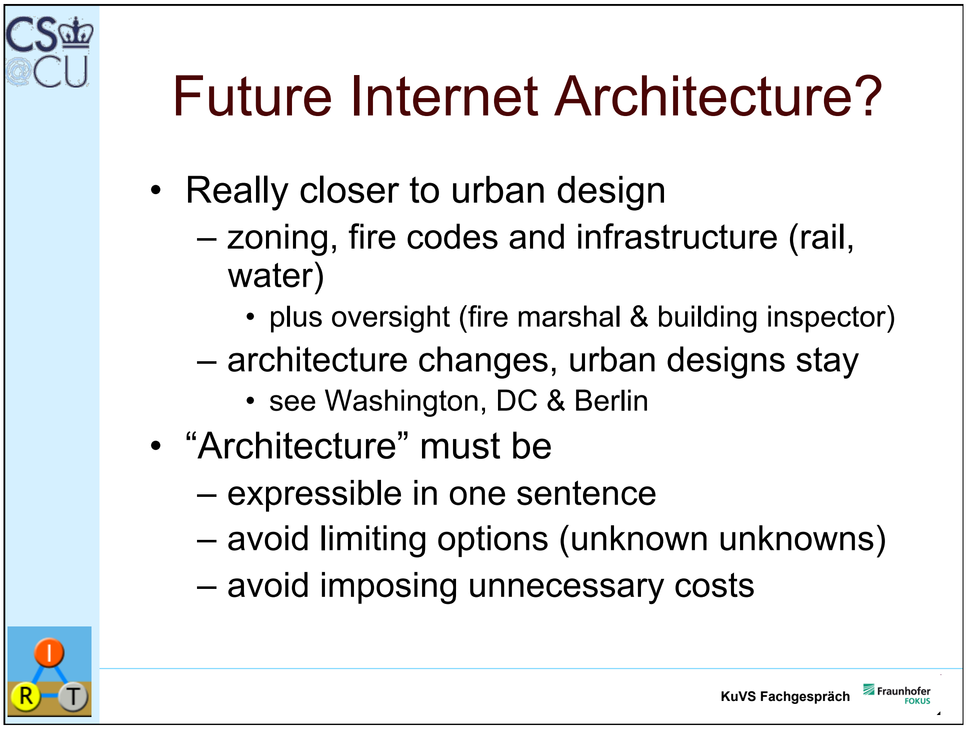
KuVS Fachgespräch Fraunhofer FOKUS



Related Work

- Cisco's Programmable Overlay Router
- Juniper's JUNOS SDK
- DaVinci project
- VROOM (virtual routers on the move)
- OpenFlow Switch
- Ethane

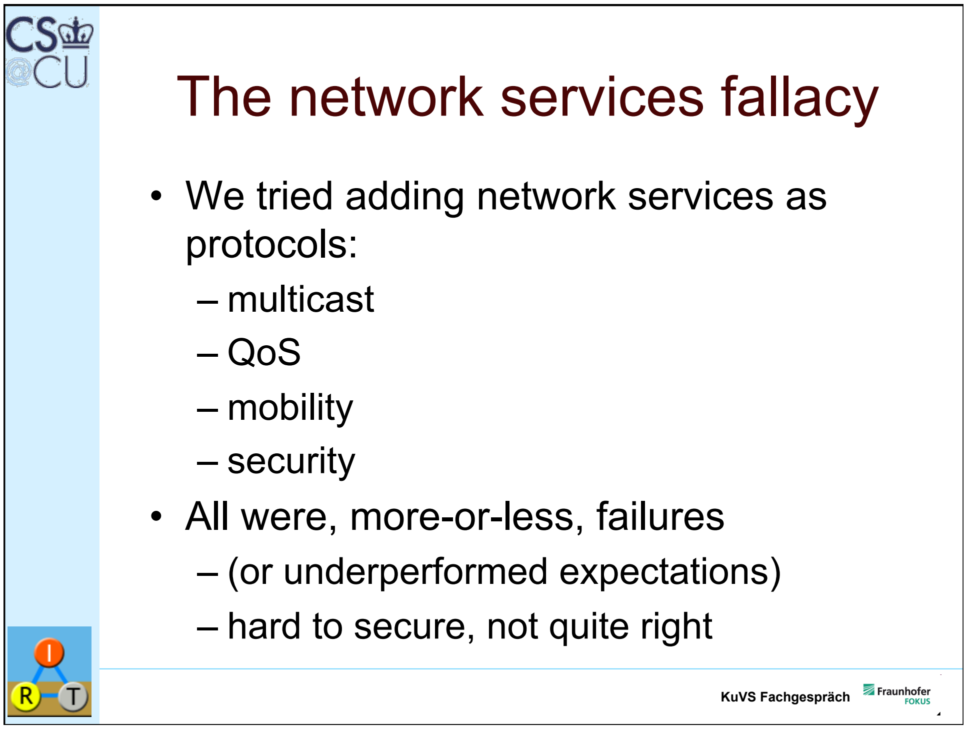
KuVS Fachgespräch



Future Internet Architecture?

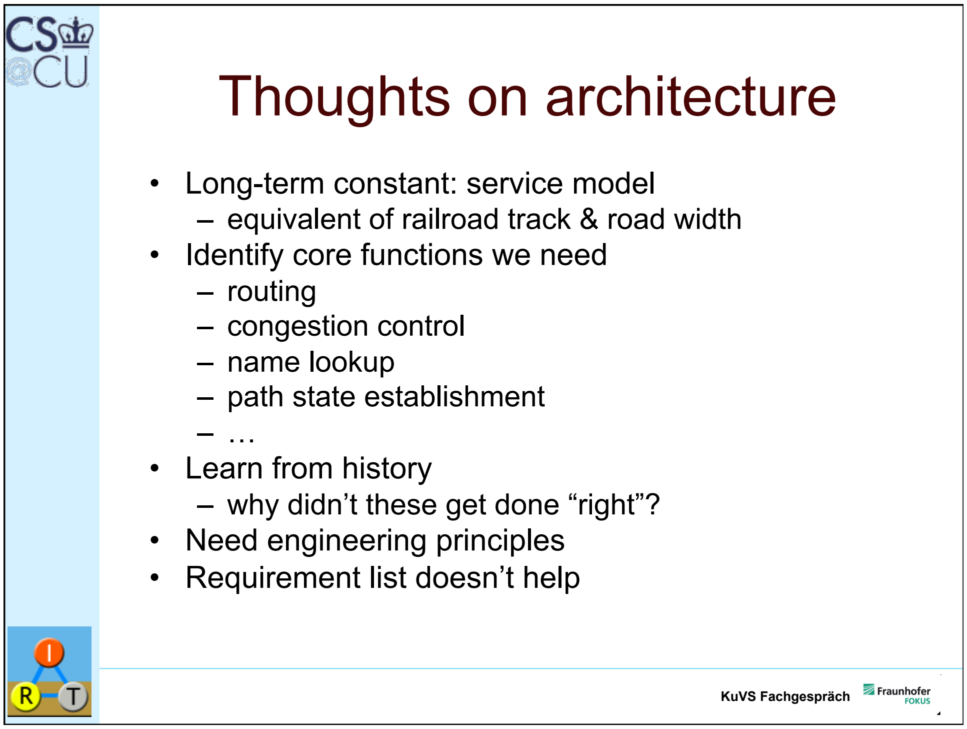
- Really closer to urban design
  - zoning, fire codes and infrastructure (rail, water)
    - plus oversight (fire marshal & building inspector)
  - architecture changes, urban designs stay
    - see Washington, DC & Berlin
- "Architecture" must be
  - expressible in one sentence
  - avoid limiting options (unknown unknowns)
  - avoid imposing unnecessary costs

KuVS Fachgespräch



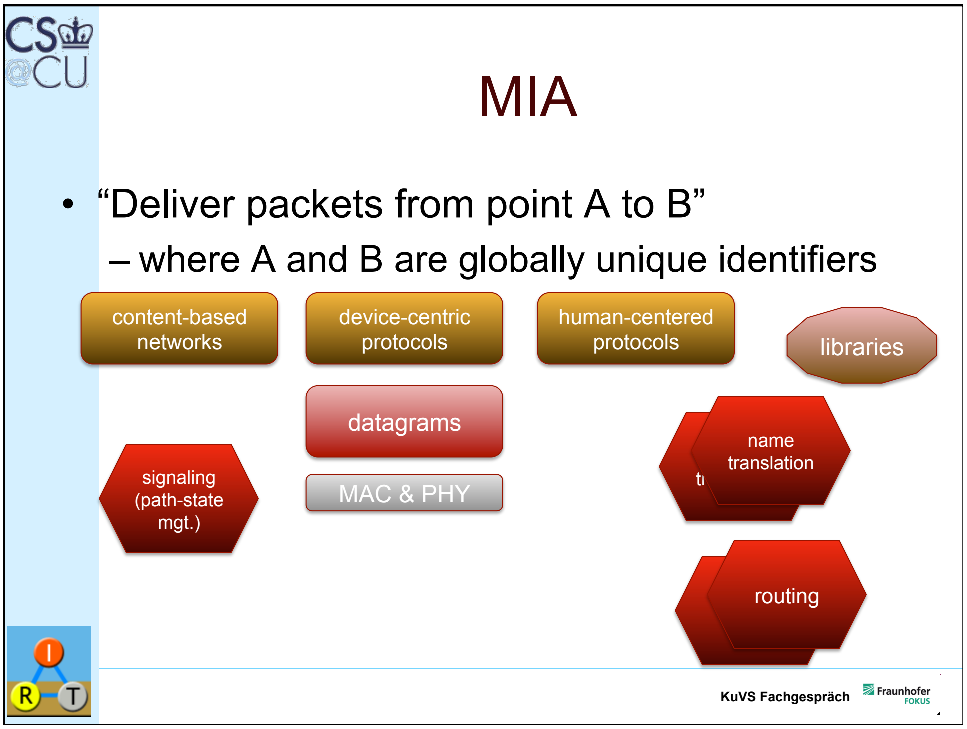
The network services fallacy

- We tried adding network services as protocols:
  - multicast
  - QoS
  - mobility
  - security
- All were, more-or-less, failures
  - (or underperformed expectations)
  - hard to secure, not quite right



Thoughts on architecture

- Long-term constant: service model
  - equivalent of railroad track & road width
- Identify core functions we need
  - routing
  - congestion control
  - name lookup
  - path state establishment
  - …
- Learn from history
  - why didn’t these get done “right”?
- Need engineering principles
- Requirement list doesn’t help



MIA

- "Deliver packets from point A to B"
  - where A and B are globally unique identifiers

content-based networks

device-centric protocols

human-centered protocols

libraries

datagrams

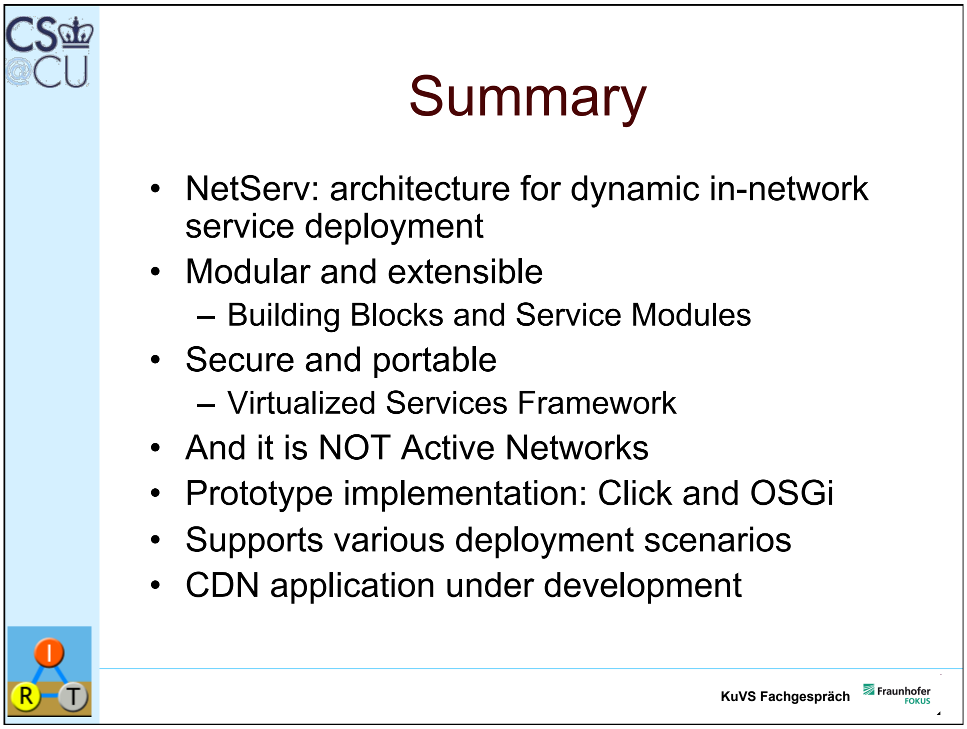
signaling (path-state mgt.)

MAC & PHY

ti

name translation

routing



Summary

- NetServ: architecture for dynamic in-network service deployment
- Modular and extensible
  - Building Blocks and Service Modules
- Secure and portable
  - Virtualized Services Framework
- And it is NOT Active Networks
- Prototype implementation: Click and OSGi
- Supports various deployment scenarios
- CDN application under development